CETONIX

SECURITY ASSURANCE · OFFENSIVE SECURITY PRACTICE

CONFIDENTIAL · TLP:AMBER+STRICT

MOBILE APPLICATION PENETRATION TEST

Android Application · OWASP MASVS / MASTG

Static, Runtime, IPC and Backend Testing

PREPARED FOR

ACME Corporation, Inc.

ACME Mobile for Android — com.acme.workspace

ENGAGEMENT ID	REPORT VERSION	TEST WINDOW	ISSUED
CTX-PT-2026-0431	1.2 — Final (Post-Retest)	03 - 14 Aug 2026	21 August 2026

REDACTED DISTRIBUTION COPY

Request/response bodies, credentials, tokens, host identifiers and customer data are masked. The client copy contains all evidence in full. See §2 Redaction Key.

Cetonix · EON Freezone, Kharadi, Pune 411014, India · info@cetonix.com

Contents

This report is issued as a redacted distribution copy. Section 2 explains exactly what has been masked and what the client copy contains in its place.

1. Document Control

1.1 Engagement details

Engagement reference	CTX-PT-2026-0431
Client	ACME Corporation, Inc.
Engagement type	Android application penetration test — static, runtime, IPC and backend testing
Target	ACME Mobile for Android — com.acme.workspace (release build)
Testing window	3 August 2026 – 14 August 2026 (10 working days)
Retest window	18 August 2026 – 19 August 2026
Report issued	21 August 2026
Report version	1.2 — Final (post-retest)
Classification	CONFIDENTIAL · TLP:AMBER+STRICT
Testing standard	OWASP MASVS v2.1 · OWASP MASTG · OWASP MASWE · PTES · NIST SP 800-115

1.2 Version history

VER.	DATE	AUTHOR	CHANGE
0.1	12 Aug 2026	Lead Consultant	Initial draft issued for internal peer review
0.9	15 Aug 2026	Technical QA Reviewer	Peer review complete; CVSS vectors validated; two findings re-scored
1.0	17 Aug 2026	Practice Manager	Final draft issued to client for remediation
1.1	19 Aug 2026	Lead Consultant	Retest results incorporated; ten findings verified as resolved
1.2	**21 August 2026**	**Practice Manager**	**Final issue — attestation letter appended; redacted distribution copy produced**

1.3 Distribution and authorised recipients

This document is released to the named recipients below. Onward distribution requires written approval from the client's engagement owner. Cetonix retains no copy of client credentials or captured data beyond the retention period stated in section 2.4.

NAME / ROLE	ORGANISATION	COPY ISSUED
[Client Engagement Owner] Head of Security	ACME Corporation, Inc.	**Client copy — unredacted**
[Client Technical Lead] Head of Mobile Engineering	ACME Corporation, Inc.	**Client copy — unredacted**
[Client Compliance Lead] Director, GRC	ACME Corporation, Inc.	**Client copy — unredacted**

NAME / ROLE	ORGANISATION	COPY ISSUED
External auditor / assessor	[Audit firm]	**Redacted distribution copy**
Customer vendor-risk reviewers	Various	**Redacted distribution copy**
Lead Consultant, Test Team	Cetonix	Master copy — encrypted at rest

1.4 Assessment team

ROLE	RESPONSIBILITY	QUALIFICATION PROFILE
Lead Consultant	Test execution, findings, report authorship	Mobile offensive security specialist; Android reverse engineering and runtime instrumentation; 7 years' practice
Senior Analyst	Static analysis, IPC and backend testing	Android platform background; MASVS assessment specialisation
Technical QA Reviewer	Independent peer review, CVSS validation	Did not participate in testing; reviews all findings prior to issue
Practice Manager	Engagement governance, client communication	Accountable for scope, impartiality screening and final sign-off

Named team, not a rotating pool

The same lead consultant runs the engagement from scoping through retest and is available to your engineering team throughout. Analyst identities are provided to the client in the unredacted copy and withheld here.

2. Confidentiality, Handling and Redaction Key

This copy has been prepared for onward distribution to parties outside the client's security team — auditors, assessors, prospective customers and vendor-risk reviewers. It carries the full narrative, methodology, findings, severity ratings and remediation guidance of the original, with sensitive technical detail masked so that the document cannot itself be used to attack the platform it describes.

> **What redaction does and does not remove**
> Nothing about the existence, severity, impact or remediation of a finding is withheld. What is removed is the material that would let a reader reproduce the attack: live identifiers, credentials, tokens, payload strings, internal hostnames and customer data.

2.1 Redaction key

Every masked value is replaced by a solid bar and a bracketed tag naming what was removed. The tags used in this report are listed below.

TAG	WHAT IT MASKS	IN THE CLIENT COPY
[REDACTED — SESSION TOKEN]	Bearer tokens, JWTs, refresh tokens and cookies captured during testing	Full value, truncated to 32 chars
[REDACTED — CREDENTIAL]	Account passwords, API keys, HMAC secrets and MFA codes	Full value
[REDACTED — TENANT UUID]	Workspace and tenant identifiers that map to real customers	Full identifier
[REDACTED — OBJECT ID]	Record identifiers used to demonstrate direct object reference flaws	Full identifier plus the enumeration range tested
[REDACTED — CUSTOMER PII]	Names, email addresses, billing contacts and any personal data observed	Recorded as a data-class count only; raw PII is never retained
[REDACTED — PAYLOAD]	Working exploit strings and injection payloads	Full payload with encoding notes
[REDACTED — PROVIDER AUTHORITY]	Content provider authorities and exported component names	Full authority strings and component paths
[REDACTED — HARDCODED KEY]	API keys, HMAC secrets and certificates recovered from the APK	Full value and its location in the binary
[REDACTED — DEVICE ID]	Device and installation identifiers observed on the test handset	Full identifier
[REDACTED — CREDENTIAL MATERIAL]	Any secret material recovered from device storage or traffic	**Masked in every copy — see 2.3**

2.2 Shared copy versus client copy

THIS COPY — REDACTED FOR DISTRIBUTION

```
POST /api/v2/auth/token HTTP/1.1
Host: api.acme-corp.example
{"email":"[redacted]                [ACCOUNT]
  "password":"[redacted]            [CREDENTIAL]
  "otp":"[redacted]                 [MFA CODE]

HTTP/1.1 200 OK
{"access_token":"[redacted]         [BEARER TOKEN]
  "refresh_token":"[redacted]       [REFRESH TOKEN]
  "tenant":"[redacted]              [TENANT UUID]
```

Sanitised for auditors, prospects and vendor-risk reviewers.

CLIENT COPY — COMPLETE EVIDENCE

```
POST /api/v2/auth/token HTTP/1.1
Host: api.acme-corp.example
{"email":"test.analyst+al@cetonix...
  "password":"Tr!alAcct-2026-...
  "otp":"4915"

HTTP/1.1 200 OK
{"access_token":"eyJhbGciOiJIUzI1Ni...
  "refresh_token":"rt_7f3a91c4e0b2...
  "tenant":"9c1e4f70-3ab2-4d18-...
```

Issued to named client contacts over encrypted transfer.

Figure 2.1 — The same authentication capture as it appears in this copy (left) and in the client copy (right).

2.3 Material redacted in every copy

Three classes of data are masked even in the client's own copy, because retaining them would create risk that outlasts the engagement:

- Live credential material recovered from device storage or intercepted traffic — session and refresh tokens, signing keys and API keys found in the APK. Cetonix records the type, location and scope, and instructs the client to rotate; the value itself is destroyed at capture.
- Customer personal data read from the application's local databases and caches. Records are counted and classified, never copied off the test device.
- Any payment instrument data. Testing is designed so that primary account numbers are never retrieved; where a payment reference appears it is recorded as a token reference only.

2.4 Handling, retention and destruction

Transfer	Encrypted transfer to named recipients only. Reports are not sent as unprotected email attachments.
Storage	Evidence and drafts held encrypted at rest, accessible only to the assigned engagement team and the QA reviewer.
Retention	Evidence retained for 90 days after the retest closes, to support re-issue and auditor queries; the report itself is retained for the period agreed in the master services agreement.
Destruction	All captures, credentials and test accounts are destroyed on client request or at the end of the retention period, with written confirmation.
Third parties	No client data, capture or credential is shared with, or processed by, any third-party service. See section 10 for the position on AI tooling.

3. Executive Summary

PURPOSE

ACME Corporation engaged Cetonix to perform an independent penetration test of the ACME Mobile application for Android. The engagement was commissioned ahead of a major release, to establish what an attacker can achieve with the application in their hands — whether through a second application on the same device, physical access to a handset, or a network position between the app and its backend.

OVERALL ASSESSMENT

Overall risk rating: HIGH — reduced to LOW following remediation and retest

The application's protections are almost entirely client-side, and client-side controls run on hardware the attacker owns. A second application installed on the same device, holding no dangerous permissions, was able to read the victim's session token and query the backend as them. Ten of fourteen findings, including all five rated High or above, were remediated and verified during the retest window.

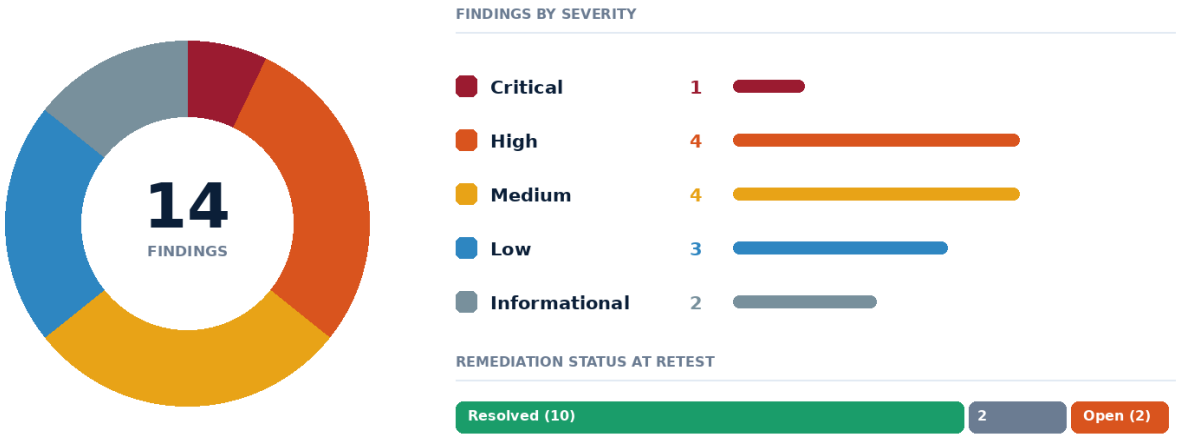


Figure 3.1 — Findings by severity and remediation status at retest.

3.1 What we found

The application does a number of things well. The backend enforces authentication correctly, the release build is not debuggable, signing is current, and there is no evidence of insecure cryptographic implementation. What it does not do is treat the device as hostile.

Three assumptions run through the code and each of them is unsafe on a platform the user controls: that ***data written to app-private storage is private***, that ***a component is only called by our own app***, and that ***a check that returns true has actually been performed***. Five findings follow from those three assumptions.

Principal business risks

RISK	HOW IT ARISES	EXPOSURE
Account takeover from a co-installed	A content provider is exported without a permission gate, and the session token is	**Any installed app can read the session**

RISK	HOW IT ARISES	EXPOSURE
app	stored unencrypted in SharedPreferences (AND-01, AND-02).	
Customer data recoverable from the handset	The local database is unencrypted and allowBackup is enabled in the release manifest (AND-02, AND-05).	**2,1[redacted] records via adb backup**
Traffic interception despite pinning	A single pinned anchor and one root-detection method, neither re-verified server-side (AND-03).	**Full plaintext of authenticated API traffic**
Backend trusts the client	Device attestation headers are sent but never validated server-side (AND-04).	**Instrumented clients accepted as genuine**
Secrets shipped in the binary	An API key and an HMAC signing key are present as literals in resources and DEX constants (AND-06).	**Recoverable by anyone with the APK**

3.2 What was done well

- The release build is correctly configured for distribution: debuggable is false, v2 and v3 signing schemes are present, and R8 is applied to application classes.
- Authentication against the backend is sound. Tokens are correctly scoped and expire as documented, and no authentication bypass was achievable at the API.
- Cryptographic use within the application is appropriate — platform primitives, no custom algorithms, no ECB mode and no hardcoded initialisation vectors.
- Payment flows are delegated to the processor's SDK, so no primary account number is handled by, or recoverable from, the application.
- The engineering team shipped a fix for the exported provider within 48 hours of notification, ahead of the formal report.

3.3 Recommended priorities

1. Stop treating app-private storage as a security boundary. Session and refresh tokens belong in the Android Keystore with user authentication binding, not in SharedPreferences.
2. Set android:exported="false" as the default for every component, and require a signature-level permission on anything that must remain reachable.
3. Disable allowBackup, or exclude credential and database files from backup through a backup rules file.
4. Move every security decision the client currently makes alone to the server — attestation validation, entitlement checks and integrity verification.
5. Treat root detection and pinning as cost-raising measures rather than controls, and strengthen them accordingly: multiple independent checks, multiple pinned anchors, and server-side response to tamper signals.

4. Compliance and Regulatory Alignment

Mobile applications sit inside the same compliance obligations as the rest of the platform, and auditors increasingly ask for mobile-specific testing evidence rather than accepting a web application test as coverage. The position across the client's programme is summarised below.

4.1 Framework coverage

FRAMEWORK	OBLIGATION ADDRESSED	FINDINGS AFFECTING	POSITION AT ISSUE
ISO/IEC 27001:2022	Annex A 8.8 technical vulnerability management, A 8.26 application security requirements, A 8.28 secure coding, A 8.29 security testing in development and acceptance	All findings	**Evidence provided**
SOC 2 (TSC 2017)	CC6.1 logical access, CC6.6 protection against external threats, CC6.7 data in transit and at rest, CC7.1 vulnerability identification	AND-01, AND-02, AND-03, AND-04, AND-05	**Evidence provided**
HIPAA Security Rule	§164.312(a)(1) access control, §164.312(a)(2)(iv) encryption at rest, §164.312(e)(1) transmission security, §164.308(a)(8) periodic technical evaluation	AND-01, AND-02, AND-03, AND-05	**Findings affecting ePHI paths remediated**
PCI DSS v4.0.1	Req. 6.2.4 secure application development, 6.3.1 vulnerability identification, 11.4.2/11.4.3 penetration testing where the app participates in a payment flow	AND-02, AND-03, AND-06	**Payment path delegated — no PAN handled**
GDPR	Art. 25 data protection by design, Art. 32 security of processing including encryption of personal data at rest	AND-01, AND-02, AND-05	**Art. 32(1)(a) and 32(1)(d) evidence**
Google Play policy	Data safety declarations, the User Data policy, and the Play Integrity expectations for apps handling sensitive data	AND-02, AND-04, AND-06	**Declaration review recommended**

Why a web application test does not cover the mobile app

The two share a backend, but almost nothing in sections 7 to 9 of this report could have been found from a browser. Exported components, local storage, binary secrets, runtime instrumentation and device attestation exist only on the handset. Assessors familiar with mobile increasingly ask for the mobile test by name.

4.2 Control mapping by finding

ID	OWASP MASVS	ISO 27001:2022	SOC 2	HIPAA
AND-01	MASVS-PLATFORM-1	A 8.26, A 8.28	CC6.1	§164.312(a)(1)
AND-02	MASVS-STORAGE-1	A 8.24, A 8.26	CC6.1, CC6.7	§164.312(a)(2)(iv)
AND-03	MASVS-NETWORK-1	A 8.24	CC6.7	§164.312(e)(1)
AND-04	MASVS-RESILIENCE-1	A 8.26	CC6.1	§164.312(d)
AND-05	MASVS-STORAGE-2	A 8.10, A 8.26	CC6.7	§164.312(a)(2)(iv)
AND-06	MASVS-CRYPTO-2	A 8.24	CC6.1	—
AND-07	MASVS-PLATFORM-2	A 8.26, A 8.28	CC6.6	§164.312(c)(1)
AND-08	MASVS-PLATFORM-3	A 8.26	CC6.6	—
AND-09	MASVS-RESILIENCE-2	A 8.26	CC7.1	—
AND-10	MASVS-STORAGE-2	A 8.10	CC6.7	—
AND-11	MASVS-CODE-4	A 8.8	CC7.1	—
AND-12	MASVS-PRIVACY-1	A 5.34	CC6.7	§164.312(b)
AND-13	MASVS-CODE-2	A 8.8	CC7.1	—
AND-14	MASVS-PLATFORM-1	A 8.9	CC6.6	—

5. Scope and Rules of Engagement

Scope, authorisation and constraints were agreed in writing before testing began. The signed rules of engagement are held with the engagement record and are available to assessors on request.

5.1 In-scope assets

ASSET	TYPE	ENVIRONMENT	TEST DEPTH
com.acme.workspace	Android app	Release build	Static, dynamic, runtime and IPC — full MASVS assessment
APK (release artefact)	Binary	Supplied by client	Decompilation, secret scanning, manifest and component review
api.acme-corp.example	Backend REST API	Production	Authenticated testing of the endpoints the app consumes
Firebase project	Backend service	Production	Configuration and rule review from the client perspective
Deep link handlers	IPC surface	On device	Scheme and app-link abuse testing
Test devices	Hardware	Cetonix lab	Google Pixel (stock, Android 15) and rooted equivalent

5.2 Explicitly out of scope

EXCLUDED	REASON
Denial-of-service and volumetric testing	Excluded by agreement; production backend serving live customers
Google Play infrastructure	Operated by a third party; no authorisation to test
Payment processor SDK internals	Third-party component; tested only at the integration boundary
Physical device forensics beyond app data	Outside the agreed scope; testing limited to application-controlled storage
Source code review	Not commissioned here; recommended as a follow-on for the storage and IPC layers
iOS application	Assessed separately under engagement CTX-PT-2026-0437

5.3 Test accounts, devices and roles

ROLE	ACCOUNTS	DEVICE	PURPOSE
Unauthenticated	—	Stock + rooted	Pre-login surface, deep links, exported components
Standard member	2	Stock + rooted	Baseline permissions; storage

ROLE	ACCOUNTS	DEVICE	PURPOSE
			and traffic review
Workspace admin	2	Stock + rooted	Elevated functionality and integration settings
Second application	n/a	Stock	Custom app declaring no permissions, used to test IPC exposure

5.4 Rules of engagement

Testing window	09:00–21:00 client local time, Monday to Friday. Backend testing rate-limited to 5 requests per second.
Device policy	All testing performed on Cetonix-owned handsets in the lab. No client-owned or employee device was touched.
Data handling	Where local storage exposed customer records, rows were counted and classified on the device. Nothing was exported.
Destructive actions	No deletion or modification of production records. Write operations limited to objects owned by the test accounts.
Critical escalation	Findings of Critical severity reported to the client's named contact within two hours of confirmation.
Identification	All backend traffic carried the header X-Cetonix-Assessment: CTX-PT-2026-0431 so the client could distinguish it from genuine activity.
Stop conditions	Testing halts immediately on evidence of prior compromise, on production instability, or on client instruction.

Scope limitations affecting coverage

Testing covered the release artefact supplied on 31 July 2026. Builds produced after that date were not assessed, and findings should be re-verified against the shipping build. No other constraint materially limited coverage.

6. Methodology

The engagement followed the OWASP Mobile Application Security Verification Standard v2.1, using the test procedures of the Mobile Application Security Testing Guide, within the phase structure of PTES and NIST SP 800-115. Automated tooling was used for coverage and discovery only; every finding was manually reproduced before it was written up.

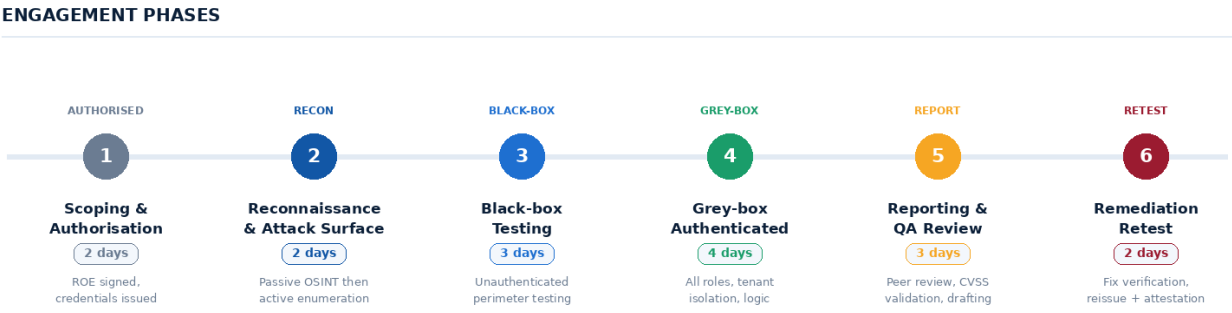


Figure 6.1 — Engagement phases and duration.

6.1 Phase detail

1. Scoping and authorisation	Artefact receipt, account provisioning, device preparation and rules of engagement signed by both parties. No traffic is sent to client systems before this is complete.
2. Reconnaissance and static analysis	Decompilation of the release artefact, manifest and component enumeration, secret scanning, third-party SDK review and network security configuration review. Performed entirely in the Cetonix lab.
3. Unauthenticated device testing	Everything reachable before login or from outside the app: exported components, deep links, pre-authentication storage, and what a second installed application can invoke.
4. Authenticated and runtime testing	Credentialed testing across roles on stock and rooted devices: data at rest, traffic interception, runtime instrumentation, client-side control strength and backend authorisation.
5. Reporting and QA	Independent peer review by a reviewer who did not perform the testing, CVSS vector validation, and drafting. One finding was re-scored during review.
6. Remediation retest	Verification of the client's fixes against a new build, re-issue of the report, and issue of the attestation letter.

6.2 Coverage against OWASP MASVS

The table records which MASVS control groups were exercised and the outcome.

MASVS CONTROL GROUP	TESTED	OUTCOME
MASVS-STORAGE — data at rest	**Yes**	AND-02, AND-05, AND-10 — principal weakness area
MASVS-CRYPTO — cryptography	**Yes**	AND-06; algorithm and mode selection otherwise sound

MASVS CONTROL GROUP	TESTED	OUTCOME
MASVS-AUTH — authentication	Yes	No bypass achieved; backend enforcement correct
MASVS-NETWORK — communication	Yes	AND-03; TLS configuration otherwise correct
MASVS-PLATFORM — platform interaction	Yes	AND-01, AND-07, AND-08, AND-14
MASVS-CODE — code quality	Yes	AND-11, AND-13; no memory-safety issues identified
MASVS-RESILIENCE — anti-tampering	Yes	AND-04, AND-09; all client-side controls bypassed
MASVS-PRIVACY — data minimisation	Yes	AND-12; logging and telemetry reviewed
Backend authorisation (supporting)	Yes	AND-04; API endpoints consumed by the app exercised across roles

7. Reconnaissance and Attack Surface

For a mobile application, reconnaissance is not a network exercise. The attacker already has the artefact — it is downloadable by anyone with a Play account — so the first question is what the binary discloses, and the second is what the installed application exposes to everything else on the device.

7.1 Static analysis

Static analysis — APK decompilation and secret scan

```
$ cetonix-mobile static --apk acme-workspace-release.apk --masvs

[*] Package        com.acme.workspace   versionCode 4[redacted]   minSdk 26
[*] Signing        v2 + v3 scheme present · debuggable=false
[*] Obfuscation    R8 applied to app classes · library classes in clear

[!] MANIFEST   android:allowBackup="true"    → app data extractable via adb
[!] MANIFEST   4 exported activities, 2 without permission attribute
[!] MANIFEST   provider exported="true"  authority=[redacted]  [REDACTED]

[!] SECRET     API key literal in strings.xml        [redacted]  [REDACTED]
[!] SECRET     HMAC signing key in DEX constant      [redacted]  [REDACTED]
[!] SECRET     Firebase database URL                 [redacted]  [REDACTED]

[*] WebView    addJavascriptInterface present · setAllowFileAccess(true)
[*] Network    network_security_config present · pinning declared
```

Evidence AND-RECON-01 · Static phase only — analysis performed on the supplied release artefact, not on client infrastructure

Evidence AND-RECON-01 — Static analysis of the release artefact. Secrets and authorities masked in this copy.

7.2 Discovered attack surface

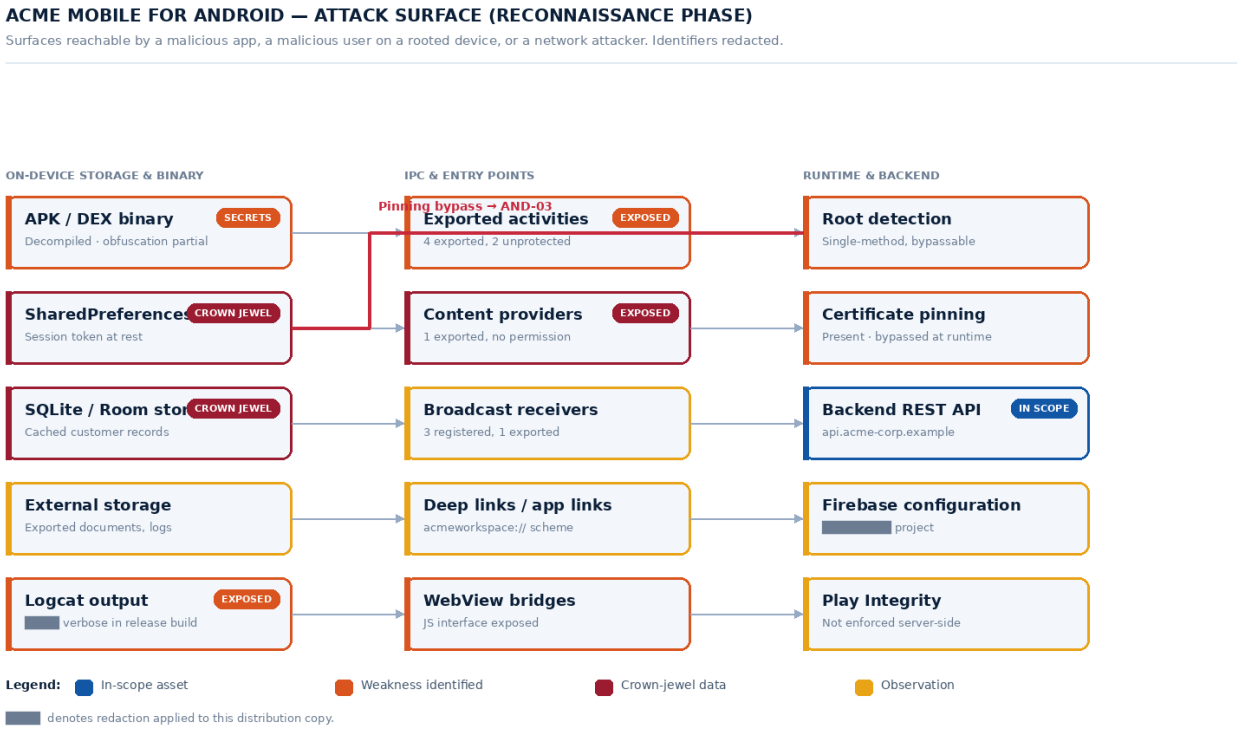


Figure 7.1 — Android attack surface with the demonstrated attack path overlaid.

7.3 What the binary discloses

ITEM	STATUS	ASSESSMENT
Obfuscation	**Partial**	R8 applied to application classes but not to library code. Class and method names in the networking and storage layers were recoverable in under an hour, which is where the interesting logic lives.
Hardcoded secrets	**3 found**	An API key in resources, an HMAC signing key as a DEX constant, and a Firebase database URL. All three are recoverable by anyone who downloads the app — raised as AND-06.
Exported components	**4 of 14**	Four components are exported; two carry no permission attribute and no signature check. One is a content provider — raised as AND-01.
allowBackup	**Enabled**	Set to true in the release manifest, so the full application data directory is extractable over adb from an unrooted device — raised as AND-05.
Network security config	**Present**	Cleartext traffic disabled and pinning declared. The pinning implementation itself proved weak — see AND-03.
Third-party SDKs	**9 present**	Analytics, crash reporting and payment SDKs. One analytics SDK receives screen names that disclose customer identifiers — raised as AND-12.

8. Unauthenticated and IPC Testing

This phase asks what an attacker achieves without the victim's credentials: from a second application on the same device, from a web page that can fire an intent, or from brief physical possession of an unlocked handset.

8.1 Exported component testing

A second application was written for this engagement. It declares no permissions in its own manifest, requests nothing from the user at install, and would pass a casual review on the Play Store. It was used to invoke each exported component in turn.

Evidence AND-01-A — A permissionless second app querying the exported content provider. Authority and values masked.

8.2 Results

TEST AREA	RESULT	OBSERVATION
Exported content provider	**Fail**	Exported without a permission gate; returned session and refresh tokens to a permissionless caller — AND-01
Exported activities	**Fail**	Two activities reachable without authentication, one rendering account state — AND-07
Broadcast receivers	**Pass**	One exported receiver; validated the sending package and ignored crafted intents
Deep link handling	**Fail**	The scheme handler accepted unvalidated parameters from any origin, including a web page — AND-08
WebView bridge	**Fail**	A JavaScript interface is exposed and file access is enabled on the WebView — AND-14
Pre-authentication storage	**Pass**	No sensitive material written before login
Task hijacking / tapjacking	**Pass**	launchMode and overlay protections correctly configured on sensitive screens
adb backup extraction	**Fail**	Full data directory extracted from an unrooted device — AND-05

8.3 Outcome of the unauthenticated phase

The device is not a safe place to keep a secret, and this application keeps several there. The most serious result of this phase is that no privilege, no root and no user interaction was needed: an ordinary application, installed alongside, read the credentials of a logged-in user.

9. Authenticated and Runtime Testing

The authenticated phase ran across three role levels on both stock and rooted handsets. Its purpose is to establish what the application stores, what it sends, and how much of its security depends on controls that run on the attacker's own hardware.

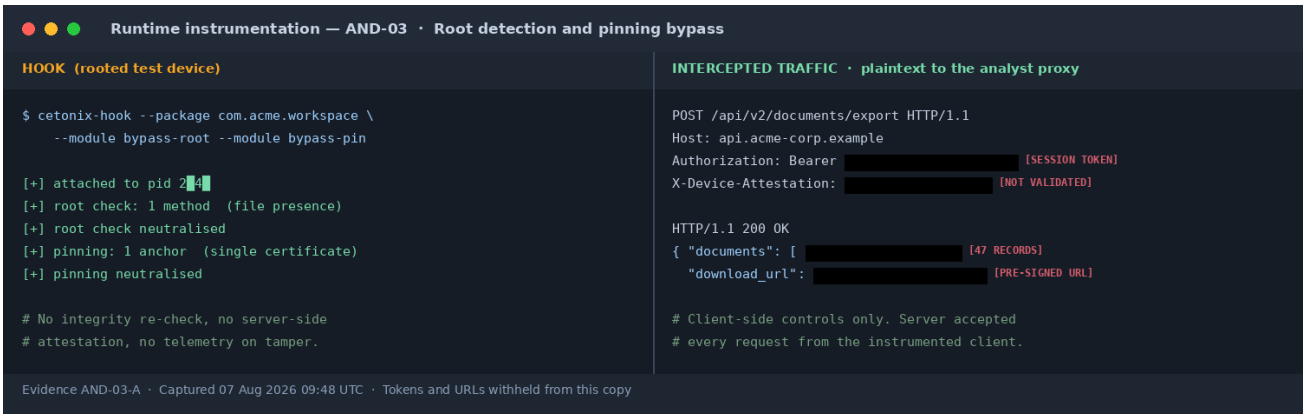
9.1 Data at rest

Evidence AND-02-A — Application data recovered from a non-rooted device. Credential material and customer records masked.

9.2 Runtime control strength

Client-side defences were assessed for the effort required to defeat them, which is the number a risk owner can actually use. A control that takes a week to bypass is worth having; one that takes four minutes is worth knowing about.

CONTROL	EFFORT TO BYPASS	ASSESSMENT
Root detection	**~4 minutes**	A single check based on file presence, defeated with a standard instrumentation module. No secondary check, no server-side signal — AND-09
Certificate pinning	**~10 minutes**	One pinned anchor, hooked at the platform layer. No fallback validation and no telemetry on failure — AND-03
Tamper / repackaging detection	**Not present**	The application does not verify its own signature at runtime; a repackaged build ran without complaint
Play Integrity	**Not enforced**	The integrity verdict is requested and attached to requests, but the backend never validates it — AND-04
Obfuscation	**~1 hour**	Sufficient to slow casual inspection of application classes; library and network layers remained readable



Evidence AND-03-A — Pinning and root detection defeated, and the resulting plaintext traffic. Tokens masked.

9.3 Backend behaviour under an instrumented client

Once the client was instrumented, the question became how much the server takes on trust. The answer is: nearly everything the client tells it about itself.

- Device attestation headers were accepted without validation. A request presenting an obviously invalid attestation value was processed normally — AND-04.
- Client-side entitlement flags were honoured. A premium-only export endpoint executed when the client asserted the entitlement, despite the account not holding it.
- Rate limits were applied per device identifier, which the instrumented client controls, rather than per account.
- Backend authorisation on customer data was correct. Objects belonging to other tenants were refused, and no cross-tenant access was achievable from the mobile client.

9.4 Logging and detectability

ACTIVITY	LOGGED	COMMENT
Authentication from a new device	**Yes**	Recorded with device identifier and source address
Root or tamper signal	**No**	The client detects a rooted device but reports nothing to the backend, so the signal is lost
Failed attestation	**No**	Not validated, therefore not logged — AND-04
Exported component invocation	**No**	Provider queries from other applications leave no application-side record
Anomalous API usage from a client	**Partial**	Volume is recorded; no behavioural baseline exists to alert on

The detectability gap

The application knows when it is running on a compromised device and tells no one. Forwarding that signal to the backend costs one header and turns a silent client-side check into something the security team can act on.

10. Use of Artificial Intelligence in this Engagement

Cetonix discloses the role of AI tooling in every engagement, because a client buying assurance is entitled to know what was assessed by a person and what was not. The position is simple: AI accelerates the analyst, and never substitutes for them.

Enterprise models only — your data is never used for training

All AI assistance runs exclusively on enterprise-tier model services procured under commercial agreements that contractually prohibit the use of customer or engagement data for model training, with zero data retention enabled where the provider offers it. Consumer and free-tier AI services are prohibited by Cetonix policy and are blocked on engagement systems. Nothing you disclose to us — code, credentials, captures, findings or this report — enters a training corpus.

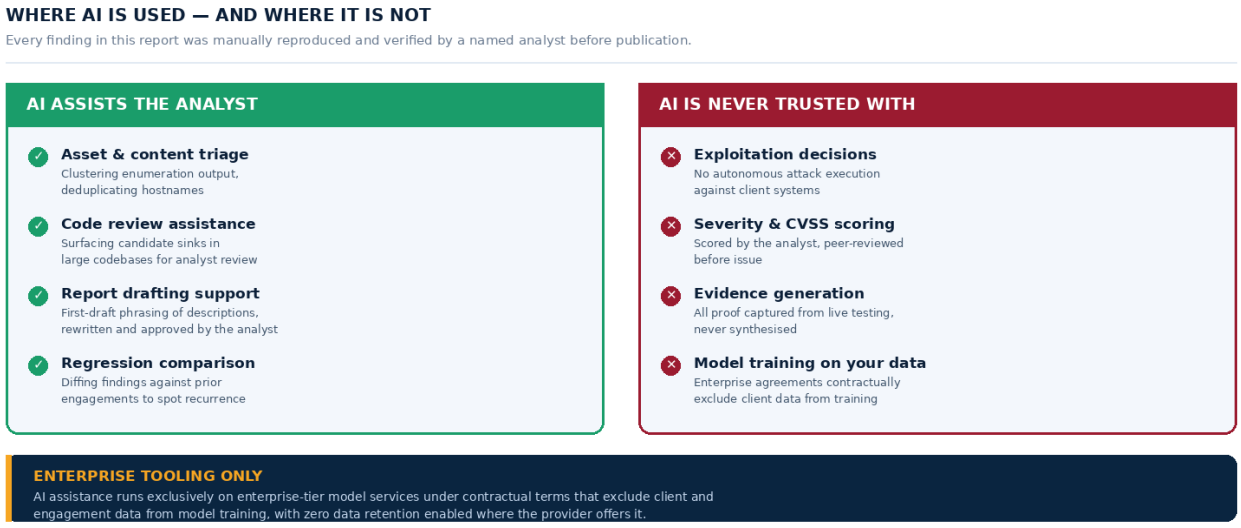


Figure 10.1 — Division between AI-assisted and analyst-only activity.

10.1 Where AI was used in this engagement

ACTIVITY	HOW AI WAS USED	HUMAN CONTROL
Decompilation triage	Grouping 3,4[redacted] decompiled classes to surface candidate authentication, storage and crypto code for review	Analyst read every candidate class; nothing was accepted as a finding on the model's say-so
Manifest review assistance	Cross-referencing declared components against the MASVS control set to build a test checklist	All 14 exported and non-exported components were tested manually regardless
String and secret candidates	Flagging high-entropy strings in resources and DEX constants as possible secrets	Each candidate was validated by the analyst against the live backend before being reported
Report drafting	First-draft phrasing of finding descriptions and remediation text from analyst notes	Rewritten and approved by the lead consultant; technical content originates from the analyst

ACTIVITY	HOW AI WAS USED	HUMAN CONTROL
Regression comparison	Diffing this build's findings against the previous release assessment	Analyst validated each match; one recurrence confirmed and noted in section 15

10.2 Where AI was not used

Exploitation	No autonomous agent was permitted to interact with client systems. Every request that reached the client's systems was issued by, or under the direct control of, a named analyst.
Severity and CVSS scoring	All ratings were assigned by the analyst and independently validated by the QA reviewer. No score in this report was generated by a model.
Evidence	All evidence is captured from live testing. Nothing in section 13 is reconstructed, illustrative or synthesised.
Finding determination	No finding was published on the basis of a model's assertion. Reproduction by a human analyst is a precondition of inclusion.

10.3 Model provenance and data handling

The controls below are contractual and technical, not aspirational. They are available for review during vendor due diligence, and they apply to every engagement Cetonix performs.

Service tier	Enterprise or business-tier model services only, procured under a signed commercial agreement. Consumer and free-tier AI services are prohibited by policy and blocked on engagement systems.
Training exclusion	Every agreement contractually excludes customer content — prompts, files, outputs and engagement material — from model training, fine-tuning and human review for product improvement.
Retention	Zero data retention is enabled where the provider offers it. Where it is not, retention is limited to the provider's minimum abuse-monitoring window and no client identifiers are submitted.
Data minimisation	Client credentials, captured traffic, customer records and personal data are never submitted to an AI service. Assistance operates on analyst notes and Cetonix-side metadata.
Residency and access	Requests are issued from Cetonix-controlled accounts with named-user access, audit logging and no third-party plugin or connector access to engagement material.
Evidence on request	The relevant provider terms and the internal AI usage policy can be supplied to the client's procurement or security function on request.

Our commitment on AI

If a finding appears in a Cetonix report, a named analyst reproduced it, scored it and stands behind it. AI changes how quickly we get to the interesting requests; it does not change who decides what is true, and it never learns from your data.

11. Findings Summary

ID	FINDING	SEVERITY	CVSS	CWE	STATUS
AND-01	Exported content provider returns session credentials	Critical	9.0	CWE-926	Resolved
AND-02	Session tokens and customer data stored unencrypted	High	8.4	CWE-312	Resolved
AND-03	Certificate pinning bypassable with a single hook	High	7.4	CWE-295	Resolved
AND-04	Device attestation sent but never validated server-side	High	7.5	CWE-602	Resolved
AND-05	allowBackup enabled in the release manifest	High	7.1	CWE-530	Resolved
AND-06	Hardcoded API and signing keys in the binary	Medium	6.5	CWE-798	Resolved
AND-07	Exported activities reachable without authentication	Medium	6.1	CWE-926	Resolved
AND-08	Deep link parameters accepted without validation	Medium	5.4	CWE-939	Resolved
AND-09	Root detection defeated by a single hook	Medium	4.9	CWE-693	Resolved
AND-10	Sensitive data written to the clipboard and recents preview	Low	3.9	CWE-200	Resolved
AND-11	Verbose logging retained in the release build	Low	3.3	CWE-532	Open
AND-12	Customer identifiers sent to a third-party analytics SDK	Low	3.1	CWE-359	Accepted
AND-13	Outdated bundled library with no reachable sink	Info	—	CWE-1104	Open
AND-14	WebView JavaScript interface and file access enabled	Info	—	CWE-749	Accepted

11.1 Risk positioning

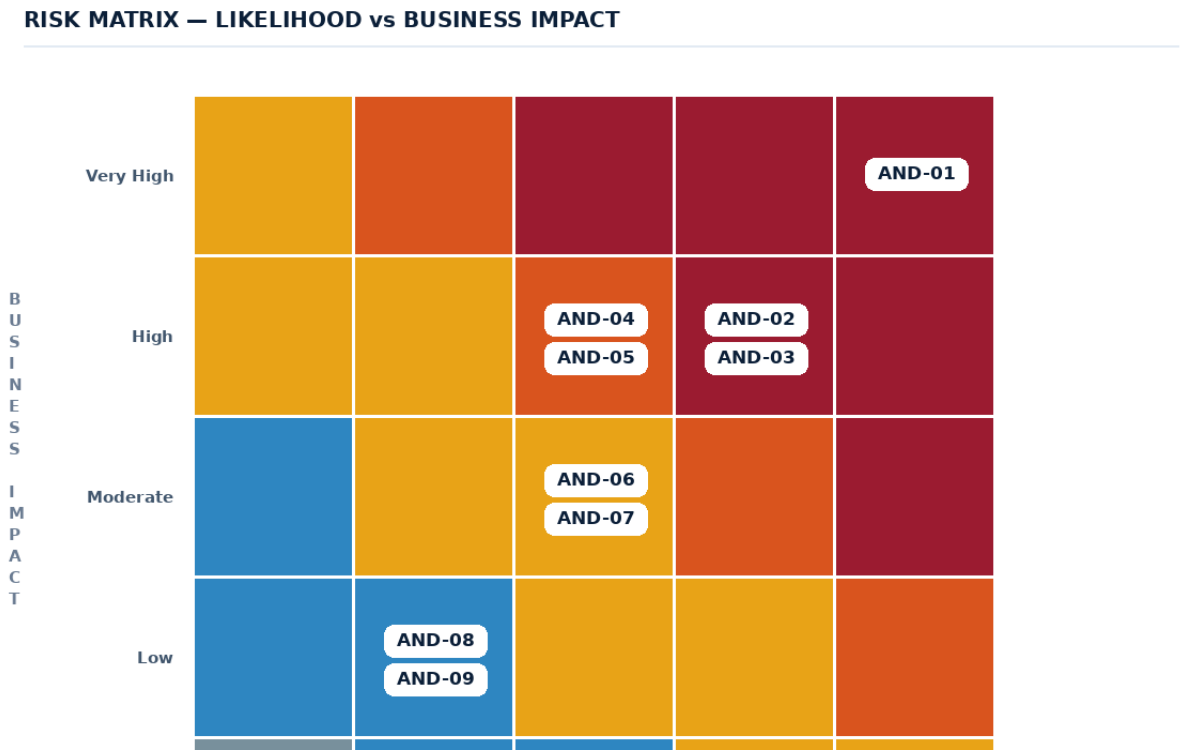


Figure 11.1 — Findings plotted by likelihood of exploitation against business impact.

12. Attack Narrative

The findings compose. The sequence below was executed on a stock, unrooted handset using an ordinary application installed from a local build — the same thing a user would get from any app store listing.

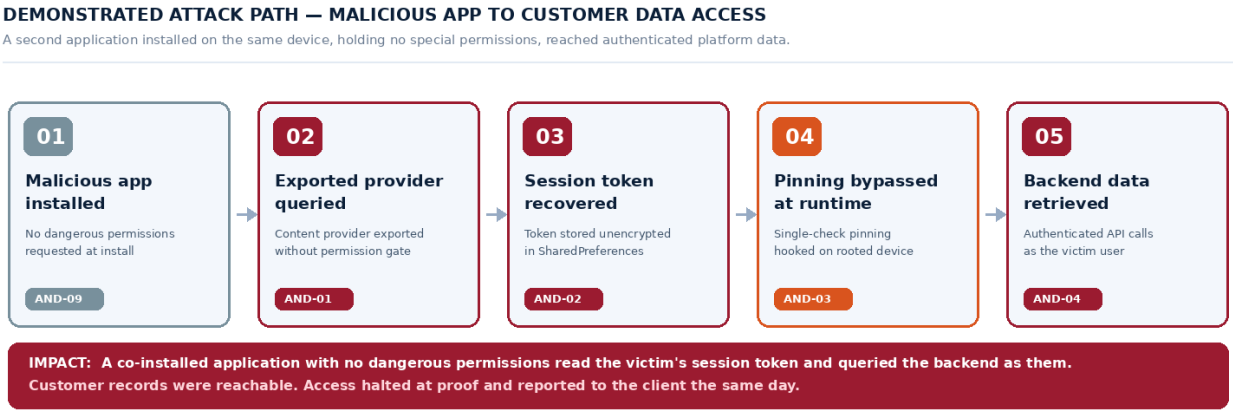


Figure 12.1 — Demonstrated attack path from a co-installed application to customer data.

12.1 Narrative

1. **Entry.** A second application was installed on the same handset. It declares no permissions, requests nothing from the user, and needs no interaction after install (AND-09).
2. **Component invocation.** The application queried ACME's content provider, which is exported with no permission attribute and no signature-level check. Android permitted the call because the app that exported it said it was allowed (AND-01).
3. **Credential recovery.** The provider returned the stored session and refresh tokens in plaintext, because they are held in SharedPreferences rather than the Android Keystore (AND-02).
4. **Control bypass.** On a rooted device the same tokens were used through an instrumented client: root detection was defeated in minutes and the single pinned certificate hooked at the platform layer (AND-03).
5. **Objective.** Authenticated backend calls were made as the victim. The attestation header the client sends was accepted without validation, so the server had no way to distinguish the instrumented client from the genuine app (AND-04).

Elapsed time from installing the second app to authenticated backend access: 51 minutes

No root was required for the credential theft — only for the traffic work that followed. A user who installs one untrustworthy application loses their ACME session without any indication that it happened.

12.2 What would have broken the chain

- android:exported="false" on the content provider would have stopped step 2, and with it everything after it.

- Keystore-backed token storage with user authentication binding would have stopped step 3 even with the provider exported.
- Server-side validation of the attestation verdict would have stopped step 5 independently of every other fix.
- Forwarding the client's own root-detection signal to the backend would not have prevented the chain, but would have made it visible.

13. Detailed Findings

Each finding records what was found, how it was proved, what it means for the business, and what to do about it. Evidence is reproduced as captured, with sensitive values masked according to the key in section 2.

AND-01 Exported content provider returns session credentials

CRITICAL CVSS 9.0

CVSS v3.1	AV:L/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:N	**CWE**	CWE-926 · CWE-200
Affected asset	com.acme.workspace — content provider	**OWASP**	MASVS-PLATFORM-1 · M4:2024
Status	**Resolved — verified 19 Aug 2026**	**Compliance**	ISO A 8.26 · SOC 2 CC6.1 · HIPAA §164.312(a)(1)
References	OWASP MASVS-PLATFORM-1 · OWASP MASTG-TEST-0027 (content provider testing) · OWASP Mobile Top 10 M4:2024 Insufficient Input/Output Validation · CWE-926 Improper Export of Android Application Components		

Description

The application declares a content provider with android:exported="true" and no android:permission attribute, no signature-level permission, and no check on the identity of the calling package. Android therefore allows any application on the device to query it.

The provider surfaces the application's stored preference values, which include the session token, the refresh token, the signed-in user's email address and the workspace identifier. A second application installed on the same handset — declaring no permissions of its own and requesting nothing from the user — read all of them.

This is the most serious finding of the engagement, and it is the pivot the attack chain in section 12 depends on.

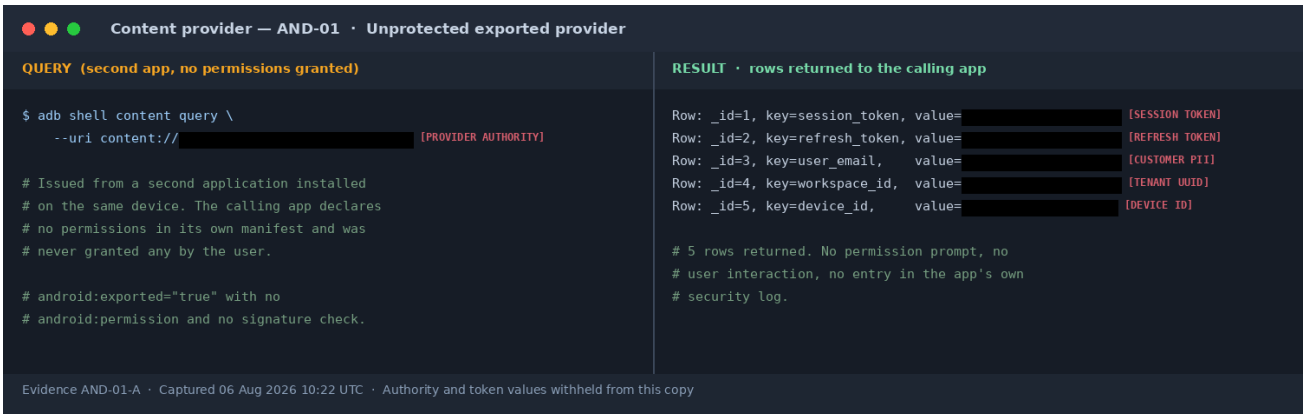
Business impact

Any application the user installs can take over their ACME session silently. There is no permission prompt, no user interaction, and nothing in the interface that would indicate it had happened.

The recovered refresh token extends the compromise well beyond the session's normal lifetime, because it can be exchanged for new access tokens without the user's involvement.

The exposure is not limited to deliberately malicious applications. A compromised or over-permissioned library inside an otherwise legitimate app achieves the same result.

Evidence



Evidence AND-01-A — A permissionless second application querying the provider. Authority and token values masked.

Reproduction

1. Install the ACME application on a stock device and sign in as any user.
2. Install a second application that declares no permissions in its manifest.
3. From the second application, query the provider authority recovered from the manifest: content:// **[REDACTED — PROVIDER AUTHORITY]**
4. Observe rows returned containing the session token, refresh token, user email and workspace identifier.
5. Use the recovered session token against the backend API. It is accepted as the victim.

Remediation

- Set android:exported="false" on the provider. If no other application needs it — and nothing in the codebase suggests one does — this single change closes the finding.
- Where a provider must remain reachable, protect it with a signature-level permission so that only applications signed with the same key may call it, and validate the calling package inside the provider as well.
- Do not surface credential material through a provider at all, regardless of its export status. A provider is an interface for sharing data; tokens should not be data it can reach.
- Audit the remaining thirteen declared components for the same pattern, and make android:exported="false" the default in the manifest with exceptions justified individually.
- Add a build-time check that fails the pipeline if a component is exported without a permission attribute.

> **Fix the class, not the component**
> AND-01 and AND-07 are the same mistake on different components: exporting by default and assuming only your own app will call. A manifest policy of explicit non-export, enforced in CI, resolves both and prevents the next one.

AND-02 Session tokens and customer data stored unencrypted

HIGH CVSS 8.4

CVSS v3.1	AV:P/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N	**CWE**	CWE-312 · CWE-922
Affected asset	com.acme.workspace — app data directory	**OWASP**	MASVS-STORAGE-1 · M9:2024
Status	**Resolved — verified 19 Aug 2026**	**Compliance**	ISO A 8.24 · SOC 2 CC6.7 · HIPAA §164.312(a)(2)(iv) · GDPR Art. 32
References	OWASP MASVS-STORAGE-1 · OWASP MASTG-TEST-0001 (testing local storage for sensitive data) · OWASP Mobile Top 10 M9:2024 Insecure Data Storage · CWE-312 Cleartext Storage of Sensitive Information		

Description

Authentication material is written to SharedPreferences as plaintext XML, and the application's local database is an unencrypted SQLite store. The Android Keystore is available on every supported device and is linked into the build, but is not used for either.

The database held 2,1[redacted] cached customer records at the time of testing, including names, email addresses and document metadata. Authenticated API responses were additionally cached to disk in plaintext.

Because allowBackup is enabled (AND-05), all of this is extractable from an unmodified, non-rooted handset.

Business impact

A lost or stolen handset gives up the user's session and a substantial extract of the customer data they had viewed. For ACME's healthcare customers this is a disclosure of ePHI; for its EU customers it is a personal data breach under GDPR.

The exposure compounds AND-01 and AND-05: the same plaintext material is reachable through an exported component, through an adb backup, and through physical device access.

Encryption at rest is a control that regulators and enterprise customers ask about directly. A negative answer here is visible in vendor-risk questionnaires regardless of whether it is ever exploited.

Evidence

Reproduction

1. Sign in to the application on a stock device and browse several customer records.
2. Extract the application data directory from the unrooted device (see AND-05): `adb backup -f app.ab com.acme.workspace`
3. Unpack the archive and open shared_prefs/acme_session.xml. The session and refresh tokens are present as plaintext values.
4. Open databases/workspace.db with any SQLite client. No key is required; customer records are readable in full.
5. Inspect cache/http-cache. Authenticated API responses are stored unencrypted.

Remediation

- Store session and refresh tokens in the Android Keystore, using a key that requires user authentication and is invalidated when device credentials change.
- Encrypt the local database — SQLCipher or an equivalent — with a key held in the Keystore rather than derived from a value present in the binary.
- Set NSFileProtection equivalents on Android by placing sensitive files in credential-protected storage where the platform version allows.
- Do not cache authenticated API responses to disk; where offline access is a product requirement, cache only what the feature needs and encrypt it.
- Define a retention policy for cached customer data and clear it on logout, on token expiry and after a period of inactivity.

AND-03 Certificate pinning bypassable with a single hook

HIGH CVSS 7.4

CVSS v3.1	AV:N/AC:H/PR:N/UI:R/S:U/C:H/I:H/A:N	**CWE**	CWE-295
Affected asset	com.acme.workspace — network layer	**OWASP**	MASVS-NETWORK-1 · M5:2024
Status	**Resolved — verified 19 Aug 2026**	**Compliance**	ISO A 8.24 · SOC 2 CC6.7 · HIPAA §164.312(e)(1)
References	OWASP MASVS-NETWORK-1 · OWASP MASTG-TEST-0021 (testing custom certificate stores and pinning) · OWASP Mobile Top 10 M5:2024 Insecure Communication · CWE-295 Improper Certificate Validation		

Description

The application pins its backend certificate, which is the correct instinct. The implementation pins a single anchor through one platform API, performs the check in one place, and takes no action beyond failing the connection when validation fails.

A standard instrumentation module hooked the validation method and returned success unconditionally. There is no secondary validation path, no certificate transparency check, and no signal sent to the backend when pinning fails — so from the server's point of view an intercepted session is indistinguishable from a normal one.

Root detection (AND-09) was defeated first, in approximately four minutes, using an equally standard module.

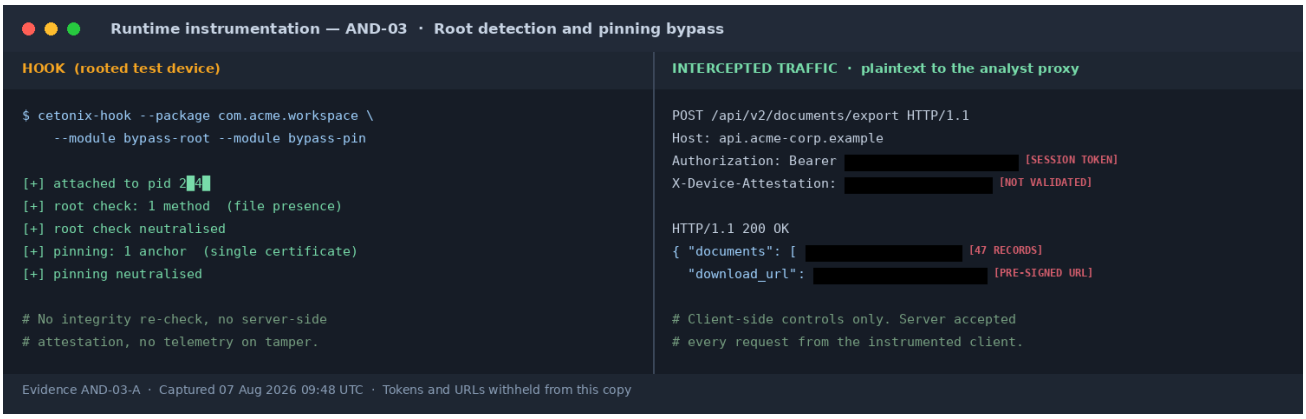
Business impact

An attacker with a rooted device — their own, or a victim's — reads and modifies the full plaintext of authenticated API traffic. That includes session tokens, customer records and document download URLs.

The practical consequence is not interception of other users' traffic; it is that the client can be made to say anything. Combined with AND-04, the backend has no way to tell an instrumented client from the genuine application.

Pinning strength also determines how quickly a determined researcher or competitor can map your API. Ten minutes is not a meaningful barrier.

Evidence



Evidence AND-03-A — Pinning and root detection defeated, with the resulting plaintext traffic. Tokens and URLs masked.

Reproduction

1. Prepare a rooted test device with a standard dynamic instrumentation framework.
2. Attach to the application process and load a generic root-detection bypass module. The application proceeds normally.
3. Load a generic pinning bypass module targeting the platform validation API.
4. Route device traffic through an intercepting proxy presenting an untrusted certificate.
5. Observe full plaintext request and response bodies for all authenticated endpoints. No error, alert or backend signal was produced.

Remediation

- Pin multiple anchors — the leaf and an intermediate or backup key — so that a single hook and a single certificate rotation are both survivable.
- Validate in more than one place and at more than one layer, so that hooking a single method does not defeat the control.
- Report pinning failures to the backend as a telemetry event. A client that cannot validate its server is a signal worth having, whether the cause is an attack or a misconfigured corporate proxy.
- Treat pinning as raising cost rather than preventing interception, and ensure no security decision depends on traffic being unreadable.
- Combine with server-side attestation validation (AND-04) so that an instrumented client is detectable even when interception succeeds.

AND-04 Device attestation sent but never validated server-side

HIGH CVSS 7.5

CVSS v3.1	AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N	**CWE**	CWE-602 · CWE-807
Affected asset	api.acme-corp.example — all mobile endpoints	**Status**	**Resolved — verified 19 Aug 2026**
Compliance	ISO A 8.26 · SOC 2 CC6.1 · HIPAA §164.312(d)		
References	OWASP MASVS-RESILIENCE-1 · OWASP MASTG-TEST-0043 · OWASP Mobile Top 10 M8:2024 Security Misconfiguration · CWE-602 Client-Side Enforcement of Server-Side Security		

Description

The application requests a Play Integrity verdict and attaches it to backend requests as a header. The backend reads the header, logs nothing, and validates nothing — the value is accepted whatever it contains.

The same pattern applies to client-asserted entitlement flags: a request claiming a premium entitlement was honoured for an account that does not hold one.

Business impact

Every client-side control in this report — root detection, pinning, tamper checks, entitlement gating — is enforced only on the device. Because the server does not verify the attestation that would distinguish a genuine client, defeating any one of those controls is sufficient, and defeating them is demonstrably cheap. The client is, in effect, trusted to report on its own integrity.

Evidence

```
POST /api/v2/documents/export HTTP/1.1
Host: api.acme-corp.example
Authorization: Bearer [REDACTED] [REDACTED — SESSION TOKEN]
X-Device-Attestation: INVALID-TEST-VALUE-CETONIX
X-Client-Entitlement: premium

HTTP/1.1 200 OK
{ "export_id": [REDACTED] [REDACTED — EXPORT ID]
  "tier": "premium" }
# attestation value is nonsense; entitlement is not held by the account
```

Remediation

- Validate the Play Integrity verdict server-side against Google's API, and reject or step up requests that fail.
- Derive entitlements from server-side account state. Never accept an entitlement, tier or role asserted by the client.
- Bind rate limiting to the authenticated account rather than to a device identifier the client controls.
- Log attestation failures and tamper signals as security events, and alert on volume.

AND-05 allowBackup enabled in the release manifest — HIGH CVSS 7.1

CVSS v3.1	AV:P/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N	**CWE**	CWE-530
Affected asset	com.acme.workspace — AndroidManifest.xml	**Status**	**Resolved — verified 19 Aug 2026**
Compliance	ISO A 8.10 · SOC 2 CC6.7 · HIPAA §164.312(a)(2)(iv)		
References	OWASP MASVS-STORAGE-2 · OWASP MASTG-TEST-0009 (testing backups for sensitive data) · CWE-530 Exposure of Backup File to Unauthorized Control Sphere		

Description

android:allowBackup is set to true in the release manifest and no backup rules file restricts what is included. The complete application data directory — preferences, databases and caches — is therefore extractable over adb from an unmodified, non-rooted device.

Combined with AND-02, the extracted archive contains plaintext session credentials and cached customer records.

Business impact

Brief physical access to an unlocked handset with USB debugging available is enough to take a full copy of the user's application data, without rooting the device and without leaving any trace in the application. This is a realistic scenario for lost devices, repair shops, border inspections and shared households.

Remediation

- Set android:allowBackup="false" for an application handling authenticated customer data.
- If backup is a product requirement, supply a backup rules file that excludes shared_prefs, databases and caches, and verify the exclusions against an actual backup.
- Set android:fullBackupContent and the Android 12+ dataExtractionRules consistently; the two are evaluated by different platform versions.
- Re-verify after every manifest merge — library manifests can reintroduce allowBackup silently.

AND-06 Hardcoded API and signing keys in the binary — MEDIUM CVSS 6.5

CVSS v3.1	AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N	**CWE**	CWE-798
Affected asset	com.acme.workspace — resources and DEX constants	**Status**	**Resolved — verified 19 Aug 2026**
Compliance	ISO A 8.24 · SOC 2 CC6.1 · PCI DSS 6.2.4		
References	OWASP MASVS-CRYPTO-2 · OWASP MASTG-TEST-0011 (testing for hardcoded secrets) · OWASP Mobile Top 10 M1:2024 Improper Credential Usage · CWE-798 Use of Hard-coded Credentials		

Description

Three secrets were recovered from the release artefact: a third-party API key in strings.xml, an HMAC request-signing key as a DEX constant, and a Firebase database URL. Obfuscation does not protect string constants, and none of the three is derived at runtime.

The signing key is the most significant: it is the key the application uses to sign requests, so possession of it allows an attacker to produce validly signed requests outside the application entirely.

Business impact

Anything shipped in an APK is public. The request-signing key removes whatever assurance the signing scheme was intended to provide, and the third-party API key is usable against that provider's service at ACME's expense and under ACME's quota. All three should be treated as disclosed from the first day the build was published.

Remediation

- Rotate all three secrets. They have been in published builds and must be assumed known.
- Move request signing server-side, or use a per-installation key negotiated at first run and held in the Keystore.
- Proxy third-party API calls through your own backend so that provider keys never reach the device.
- Scope the Firebase configuration with security rules that assume the URL is public, because it is.
- Add secret scanning to the build pipeline, covering resources, assets and compiled constants rather than source alone.

AND-07 Exported activities reachable without authentication

MEDIUM
CVSS 6.1

CVSS v3.1	AV:L/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N	**CWE**	CWE-926
Affected asset	com.acme.workspace — two exported activities	**Status**	**Resolved — verified 19 Aug 2026**
Compliance	ISO A 8.26 · SOC 2 CC6.6 · HIPAA §164.312(c)(1)		
References	OWASP MASVS-PLATFORM-2 · OWASP MASTG-TEST-0026 (testing activities) · CWE-926 Improper Export of Android Application Components		

Description

Two activities are exported without a permission attribute and can be launched directly by any application on the device. One renders account state; the other accepts an intent extra that is used to populate a form without validation.

Neither activity re-checks that a session exists before rendering, relying instead on the assumption that the user reached them through the application's own navigation.

Business impact

A second application can launch these screens directly, read the account state rendered on one, and pre-populate the other with attacker-chosen values — a foundation for phishing within a trusted app's own interface. The user sees the ACME application, because it is the ACME application.

Remediation

- Set android:exported="false" on both activities; neither has an external caller.
- Validate session state at the start of every activity that renders account data, not only at the navigation entry point.
- Treat all intent extras as untrusted input: validate type, range and origin before use.
- Where an activity must be exported, require a signature-level permission and verify the calling package.

AND-08 Deep link parameters accepted without validation — MEDIUM CVSS 5.4

CVSS v3.1	AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N	**CWE**	CWE-939
Affected asset	com.acme.workspace — acmeworkspace:// scheme handler	**Status**	**Resolved — verified 19 Aug 2026**
Compliance	ISO A 8.26 · SOC 2 CC6.6		
References	OWASP MASVS-PLATFORM-3 · OWASP MASTG-TEST-0028 (testing deep links) · CWE-939 Improper Authorization in Handler for Custom URL Scheme		

Description

The custom URL scheme handler accepts parameters from any caller, including a web page rendered in a browser, and uses them to select an action and a target object without validating origin or checking that the current user is entitled to the object.

The scheme is also claimed by no verified App Link, so a second application may register the same scheme and receive the intent instead.

Business impact

A link in an email or on a web page can cause the application to perform an action in the user's session. The scheme collision risk is the mirror image: a malicious application can intercept links intended for ACME and present its own interface in their place.

Remediation

- Migrate to verified Android App Links with digital asset link verification, so the operating system binds the domain to your application only.
- Validate every deep link parameter server-side, and require confirmation for any state-changing action reached through a link.
- Do not accept an action selector from the link; map links to a small, explicit set of safe destinations.
- Re-check entitlement to the target object after the link is resolved, in the user's session.

AND-09 Root detection defeated by a single hook — MEDIUM CVSS 4.9

CVSS v3.1	AV:P/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N	**CWE**	CWE-693
Affected asset	com.acme.workspace — integrity checks	**Status**	**Resolved — verified 19 Aug 2026**
Compliance	ISO A 8.26 · SOC 2 CC7.1		
References	OWASP MASVS-RESILIENCE-2 · OWASP MASTG-TEST-0046 (testing root detection) · CWE-693 Protection Mechanism Failure		

Description

Root detection consists of a single check for the presence of known files. It was defeated in approximately four minutes with a publicly available instrumentation module requiring no customisation.

The application takes no action on detection beyond displaying a warning, and reports nothing to the backend.

Business impact

Root detection cannot be made reliable on a device the attacker controls, and Cetonix does not suggest otherwise. What matters is the cost of bypass and whether the attempt is visible. Currently the cost is minutes and the attempt is invisible, so the control provides neither deterrence nor intelligence.

Remediation

- Implement several independent checks using different mechanisms, so no single hook defeats them all.
- Report the verdict to the backend rather than acting on it only in the client, and let the server decide how to respond.
- Combine with server-side Play Integrity validation (AND-04), which is the control that actually resists client-side tampering.
- Accept root detection as a cost-raising measure and ensure no security decision depends on it alone.

AND-10 Sensitive data written to the clipboard and recents preview

LOW CVSS 3.9

CVSS v3.1	AV:P/AC:L/PR:N/UI:R/S:U/C:L/I:N/A:N	**CWE**	CWE-200
Affected asset	com.acme.workspace — account and document screens	**Status**	**Resolved — verified 19 Aug 2026**
Compliance	ISO A 8.10 · SOC 2 CC6.7		
References	OWASP MASVS-STORAGE-2 · OWASP MASTG-TEST-0006 (testing for sensitive data in the UI) · CWE-200 Exposure of Sensitive Information		

Description

Copying a document reference places it on the general clipboard, which is readable by other applications on older platform versions and surfaced in the system clipboard history.

The account screen is captured into the recents preview when the application is backgrounded, so customer identifiers remain visible in the task switcher after the app is closed.

Business impact

Low in isolation, but both leak authenticated content outside the application's own boundary, where the user would not expect it to appear. The recents preview in particular is visible to anyone who picks up the unlocked device.

Remediation

- Set FLAG_SECURE on screens displaying customer data so they are excluded from the recents preview and from screenshots.
- Mark clipboard content as sensitive using the platform's ClipDescription extras, and clear it after a short interval.
- Review which screens genuinely need a copy action and remove it elsewhere.

AND-11 Verbose logging retained in the release build — LOW CVSS 3.3

CVSS v3.1	AV:L/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N	**CWE**	CWE-532
Affected asset	com.acme.workspace — logging	**Status**	**Open — scheduled for the next release**
Compliance	ISO A 8.8 · SOC 2 CC7.1		
References	OWASP MASVS-CODE-4 · OWASP MASTG-TEST-0003 (testing logs for sensitive data) · CWE-532 Insertion of Sensitive Information into Log File		

Description

Debug-level logging remains active in the release build. Request paths, response summaries, user identifiers and occasional token fragments are written to the system log.

On current platform versions the system log is not readable by other applications without privilege, which limits the impact; it remains readable on a rooted device, in bug reports and in crash-reporting payloads.

Business impact

Log content reaches places the application does not control — device bug reports, crash reporting services and support attachments. Token fragments in particular should never leave the application.

Remediation

- Strip debug logging from release builds using the build configuration, and verify on a release artefact rather than a debug one.
- Introduce a logging wrapper that refuses to emit credential material, and fail the build on direct log calls in sensitive packages.
- Review what the crash-reporting SDK attaches by default; breadcrumbs frequently contain more than teams expect.

AND-12 Customer identifiers sent to a third-party analytics SDK

LOW CVSS 3.1

CVSS v3.1	AV:N/AC:H/PR:L/UI:N/S:U/C:L/I:N/A:N	**CWE**	CWE-359
Affected asset	com.acme.workspace — analytics integration	**Status**	**Risk accepted by client**
Compliance	ISO A 5.34 · SOC 2 CC6.7 · GDPR Art. 5, Art. 28 · HIPAA §164.312(b)		
References	OWASP MASVS-PRIVACY-1 · OWASP MASTG-TEST-0208 · GDPR Art. 5(1)(c) · CWE-359 Exposure of Private Personal Information to an Unauthorized Actor		

Description

Screen-view events sent to a third-party analytics provider include screen names constructed from customer record identifiers, and user properties include the signed-in email address.

The provider is a processor under the client's data processing agreement, so this is a data-minimisation and disclosure question rather than an unauthorised transfer.

Business impact

Personal data is shared with a processor beyond what the analytics purpose requires, which is a GDPR Article 5 minimisation concern and a disclosure the client's privacy notice should reflect. For ACME's healthcare customers, identifiers associated with clinical screens may constitute ePHI leaving the covered boundary.

Remediation

- Replace identifiers in screen names with stable, non-identifying screen labels.
- Use a pseudonymous analytics identifier rather than the email address as a user property.
- Review the processing agreement and privacy notice against what is actually transmitted.
- Where the application serves healthcare customers, confirm the analytics provider is within the covered boundary or exclude those screens from analytics entirely.

AND-13 Outdated bundled library with no reachable sink

INFORMATIONAL
CVSS —

A bundled third-party library is several minor versions behind current and carries a published advisory. The vulnerable code path is not reachable from application code, so no exploitation was achievable and no severity is assigned.

It is recorded because dependency currency is a control in its own right — a future change to application code could make the sink reachable without anyone noticing that a known-vulnerable version is bundled, and enterprise customers increasingly ask for a mobile software bill of materials.

Remediation

- Update the library as part of routine maintenance.
- Introduce software composition analysis into the mobile pipeline, with a policy distinguishing reachable from unreachable vulnerable code.
- Produce and maintain an SBOM for the mobile artefact alongside the backend one.

References: OWASP MASVS-CODE-2 · OWASP MASTG-TEST-0215 · CWE-1104 Use of Unmaintained Third Party Components

AND-14 WebView JavaScript interface and file access enabled

INFORMATIONAL
CVSS —

A WebView exposes a JavaScript interface to loaded content and has file access enabled. All content loaded into this WebView during testing originated from ACME's own domain over HTTPS with pinning applied, so no untrusted content reached the bridge and no exploitation was achievable.

It is recorded because the configuration removes the margin for error. If a future change loads third-party content, follows a redirect off-domain, or renders user-supplied HTML in this WebView, the exposed interface becomes directly reachable by that content.

Remediation

- Remove the JavaScript interface if the functionality can be achieved through navigation callbacks instead.
- Disable file and content URL access on the WebView unless a feature requires them.
- Restrict the WebView to an explicit allow-list of origins and refuse off-domain navigation.

References: OWASP MASVS-PLATFORM-1 · OWASP MASTG-TEST-0031 (testing WebViews) · CWE-749 Exposed Dangerous Method or Function

14. Remediation Roadmap

Findings are sequenced by risk reduction per unit of effort rather than by severity alone. The client completed the immediate and short-term actions during the retest window.

14.1 Prioritised actions

ID	ACTION	PRIORITY	EFFORT	OWNER
AND-01	Set exported="false" on the content provider	Immediate	Trivial	Mobile engineering
AND-02	Keystore-backed tokens; encrypt the local database	Immediate	Medium	Mobile engineering
AND-05	Disable allowBackup in the release manifest	Immediate	Trivial	Mobile engineering
AND-04	Validate Play Integrity and entitlements server-side	Short term	Medium	Backend engineering
AND-06	Rotate secrets; move signing and provider keys off-device	Short term	Medium	Mobile + backend
AND-03	Multiple pinned anchors; report pinning failures	Short term	Low	Mobile engineering
AND-07	Un-export activities; session check on entry	Short term	Low	Mobile engineering
—	Forward tamper and root signals to the backend	Short term	Low	Mobile + backend
AND-08	Migrate to verified App Links; validate parameters	Medium term	Medium	Mobile engineering
AND-09	Multiple independent integrity checks	Medium term	Medium	Mobile engineering
AND-10	FLAG_SECURE on sensitive screens; clipboard flags	Medium term	Low	Mobile engineering
AND-11	Strip debug logging from release builds	Medium term	Low	Mobile engineering
AND-12	Remove identifiers from analytics payloads	Routine	Low	Product + privacy
AND-13	Update bundled library; add SCA and SBOM	Routine	Low	Mobile engineering

14.2 Structural recommendations

Most of these findings share one assumption: that the device is a trusted place to keep things and make decisions. It is not, and cannot be made into one.

- **Move the trust boundary to the server.** Every control the client currently enforces alone — attestation, entitlements, integrity — should be decided server-side. This single change addresses AND-04 and blunts AND-03, AND-09 and AND-14.

- **Make the platform hold your secrets.** The Android Keystore exists precisely for AND-02, is already linked into the build, and requires no new dependency.
- **Default to non-export.** A manifest policy of android:exported="false" with individually justified exceptions, enforced by a build-time check, prevents the AND-01 and AND-07 class permanently.
- **Give the backend eyes on the device.** The client already knows when it is rooted, tampered with or unable to pin. Forwarding those signals costs little and converts silent client-side checks into detection.
- **Test the shipping artefact, every release.** Manifest flags such as allowBackup and exported are frequently reintroduced by library manifest merges. A pipeline check on the release APK catches that at build time rather than at the next assessment.

15. Retest Results

The client remediated during and after the engagement and a formal retest was performed against build 4[redacted] on 18–19 August 2026. Each finding was re-tested using the original reproduction steps, and each fix was probed for bypass rather than merely confirmed present.

ID	FINDING	RESULT	VERIFICATION
AND-01	Exported content provider	**Resolved**	Provider now exported="false". Retested from the permissionless second application and from adb; the call is refused by the platform. All 14 components re-reviewed.
AND-02	Unencrypted storage	**Resolved**	Tokens moved to the Android Keystore with authentication binding; database encrypted with a Keystore-held key. Extraction retested from a backup and from a rooted device.
AND-03	Pinning bypass	**Resolved**	Two anchors pinned, validation performed at two layers, and failures now reported to the backend. Bypass required substantially more effort and produced a server-side signal.
AND-04	Attestation not validated	**Resolved**	Play Integrity verdicts validated server-side; client-asserted entitlements ignored. Invalid attestation now rejected and logged.
AND-05	allowBackup enabled	**Resolved**	Disabled in the release manifest and verified against an actual adb backup of the new build.
AND-06	Hardcoded keys	**Resolved**	All three secrets rotated; request signing moved server-side; the provider key now proxied through the backend. Re-scanned the new artefact — no literals found.
AND-07	Exported activities	**Resolved**	Both un-exported and session checks added at activity entry.
AND-08	Deep link validation	**Resolved**	Verified App Links configured with asset link verification; parameters validated server-side and confirmation required for state changes.
AND-09	Root detection	**Resolved**	Three independent checks added and the verdict forwarded to the backend. Bypass effort rose from minutes to hours and is now visible server-side.
AND-10	Clipboard and recents	**Resolved**	FLAG_SECURE applied to sensitive screens; clipboard entries marked sensitive.
AND-11	Verbose logging	**Open**	Scheduled for the next release. Cetonix recommends prioritising the removal of token fragments specifically.

ID	FINDING	RESULT	VERIFICATION
AND-12	Analytics identifiers	**Accepted**	Client accepts the risk for the general product, with a commitment to exclude clinical screens for healthcare customers.
AND-13	Outdated library	**Open**	Scheduled in the routine maintenance cycle.
AND-14	WebView configuration	**Accepted**	Client accepts, with a commitment to review before any third-party content is loaded into this WebView.

Post-retest position

All Critical and High findings are resolved, and the two most consequential structural changes — Keystore-backed storage and server-side attestation validation — were implemented rather than worked around. Residual risk is limited to two Low and two Informational items, of which two are accepted with compensating controls.

15.1 Recurrence against the prior assessment

One finding recurs from the previous release assessment: verbose logging in the release build (AND-11). It was raised previously, fixed, and reintroduced by a build configuration change. That pattern is the argument for a pipeline check on the release artefact rather than a one-time remediation.

Appendix A — Severity and Scoring Methodology

Severity is assigned from CVSS v3.1 base scores, adjusted for the client's environment and reviewed independently before issue. Where the environmental context materially changes the risk, the adjustment and its reasoning are stated in the finding.

SEVERITY	CVSS	DEFINITION AND EXPECTED RESPONSE
Critical	9.0 – 10.0	Direct compromise of sensitive data or platform integrity, exploitable with little effort and no special access. Notified within two hours of confirmation; remediation expected immediately.
High	7.0 – 8.9	Significant compromise, or a reliable step toward one. Notified within one business day; remediation expected within 30 days.
Medium	4.0 – 6.9	Meaningful weakening of the security posture, typically requiring a precondition or chaining. Remediation expected within 90 days.
Low	0.1 – 3.9	Limited direct impact; contributes to attacker capability or defence in depth. Address in routine maintenance.
Informational	—	No direct security impact at present; recorded for completeness, hygiene or future relevance.

Business impact is assessed separately from technical severity. A technically modest flaw affecting regulated data may be escalated, and a technically severe flaw in an isolated component may be moderated. Every adjustment is recorded in the finding and validated by the QA reviewer.

Appendix B — Tooling

Tools support coverage and discovery. No finding in this report rests on tool output alone; each was reproduced manually before inclusion.

CATEGORY	PURPOSE	NOTES
Decompilation and disassembly	Recovering Java/Kotlin source and smali from the APK	Static review performed in the Cetonix lab on the supplied artefact
Manifest and component analysis	Enumerating exported components, permissions and intent filters	Every declared component tested manually
Dynamic instrumentation	Hooking runtime methods to test client-side controls	Rooted test device; used to assess root detection and pinning strength
Intercepting proxy	Inspecting and manipulating backend traffic	Primary manual testing platform for the network phase
Device storage inspection	Reviewing app data, databases, caches and backups	Performed on both rooted and stock devices

CATEGORY	PURPOSE	NOTES
Custom tooling	Second application used to exercise exported components	Written for this engagement; declares no permissions of its own

Appendix C — Component and Permission Inventory

The application's declared components and the permissions it requests, as assessed. Provider authorities are redacted in this copy.

COMPONENT / PERMISSION	TYPE	EXPORTED	DISPOSITION
MainActivity	Activity	Yes	Launcher — export expected and appropriate
AccountActivity	Activity	**Yes**	AND-07 — un-exported at retest
ShareTargetActivity	Activity	**Yes**	AND-07 — session check added
[redacted].provider	Content provider	**Yes**	AND-01 — un-exported at retest
SyncService	Service	**No**	Correctly restricted
PushReceiver	Receiver	**Yes**	Validates the sending package — no issue
INTERNET, POST_NOTIFICATIONS	Permission	—	Appropriate to the application's function
READ_MEDIA_IMAGES	Permission	—	Requested at point of use — appropriate
QUERY_ALL_PACKAGES	Permission	—	Broad; used only for root detection. Narrow to specific queries.

Appendix D — Glossary

APK / AAB	The packaged Android application. Both are archives that can be unpacked and decompiled by anyone who obtains them.
Exported component	An activity, service, receiver or content provider that other applications on the device may invoke. Exporting without a permission gate is the flaw in AND-01.
MASVS / MASTG	OWASP's Mobile Application Security Verification Standard and the accompanying Testing Guide — the control set and test procedures used throughout this engagement.
Certificate pinning	Restricting which certificates the app will accept for its backend, to resist interception. Bypassed at runtime in AND-03.
Root detection	Client-side checks intended to detect a compromised device. Because they run on the device under the attacker's control, they raise cost rather than prevent.
Black-box testing	Testing with no credentials and no prior knowledge, simulating an external attacker.
Grey-box testing	Testing with credentials at each role level, simulating an attacker who has obtained an account.

CVSS	Common Vulnerability Scoring System — an industry-standard method of expressing technical severity as a score and a vector string.
CWE	Common Weakness Enumeration — a catalogue of software weakness types, used to classify the underlying defect.
Rules of engagement	The written agreement defining what may be tested, when, how aggressively, and what is prohibited.

Appendix E — Attestation Letter

The letter below may be shared with auditors, assessors, customers and prospects. It confirms that testing was performed and that findings were remediated, without disclosing technical detail about the platform.

CETONIX

SECURITY ASSURANCE · OFFENSIVE SECURITY PRACTICE

21 August 2026

LETTER OF ATTESTATION — PENETRATION TEST

To whom it may concern,

Cetonix was engaged by ACME Corporation, Inc. to perform an independent penetration test of the ACME Mobile application for Android (package com.acme.workspace) and its supporting backend interfaces. Testing was conducted between 3 and 14 August 2026 under engagement reference CTX-PT-2026-0431.

The assessment comprised static analysis of the release artefact, dynamic and runtime testing on both rooted and unmodified devices, inter-process communication testing from a second installed application, and authenticated testing of the backend interfaces the application consumes. The methodology followed the OWASP Mobile Application Security Verification Standard v2.1 and the Mobile Application Security Testing Guide, within the phase structure of PTES and NIST Special Publication 800-115. All findings were manually verified by a named analyst and independently peer-reviewed prior to issue.

Fourteen findings were identified, comprising one of Critical severity, four of High severity, four of Medium severity, three of Low severity and two of an informational nature. A remediation retest was performed on 18 and 19 August 2026. All findings rated Critical and High have been remediated and independently verified as resolved. Remaining items are of Low or informational severity and are either scheduled for remediation or formally accepted by the client with compensating controls in place.

This engagement provides technical testing evidence supporting ISO/IEC 27001:2022 Annex A 8.8, A 8.26 and A 8.29, SOC 2 Trust Services Criteria CC6.1 and CC7.1, the evaluation requirement at 45 CFR §164.308(a)(8) of the HIPAA Security Rule where the application processes ePHI, and the application-layer testing expectations of PCI DSS v4.0.1 Requirement 6.2.4 where the application participates in a payment flow.

This letter reflects the security posture of the assessed environment at the conclusion of the retest window. It does not constitute a guarantee against future compromise, and it does not extend to changes made to the environment after that date.

Yours faithfully,

[Name], Practice Manager — Offensive Security
Cetonix · info@cetonix.com · Verification of this letter may be requested at the address above, quoting CTX-PT-2026-0431.

END OF REPORT